1-1-2005

PeerCredential: a support reputation-based trust framework for peer-to-peer applications

William Donald Sears
Iowa State University

PeerCredential: A support reputation-based trust framework for peer-to-peer applications

by

William Donald Sears

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Co-majors: Information Assurance; Computer Engineering

Program of Study Committee:
Yong Guan, Co-major Professor
Thomas Daniels, Co-major Professor
Clifford Bergman

Iowa State University

Ames, Iowa

2005

Graduate College
Iowa State University

This is to certify that the master's thesis of

William Donald Sears

has met the thesis requirements of Iowa State University

DEDICATION

I would like to dedicate this thesis to my wife Leona, my parents John and Margo, and my sister Lisa.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Distributed environments, like Peer-to-Peer and scientific networks, often require those that use the system (i.e. users) to utilize other nodes (i.e. resources) that are unknown. Users are unable to identify whether a resource will be honest, selfish, or malicious. In order for users to reduce the risk of using a malicious resource and to motivate nodes from being selfish, we propose an adaptive reputation-based trust framework for distributed system applications. In this framework, users are able to appraise a resource by using *price* and a quantifiable metric of *trust* that is gathered from its own view and the views of other peers (i.e. references) regarding the reputation of the resource. The appraisal process provides the user with a reliable metric that can be used in the process of resource scheduling and selection. In response, economic resources will compete with each other using a variety of strategies that attempt to maximize their profit. The simulation results show that the framework is user friendly by providing long-term high satisfaction, filters out malicious nodes, and encourages resources to provide reliable and high-quality service.

CHAPTER 1. OVERVIEW

In this document, we will show that reputation, trust, and incentive can be correlated to provide an optimal selection of a resource in a peer-to-peer (P2P) environment.

1.1 Introduction

In distributed P2P applications, we can differentiate the roles of peers as *user* nodes and *resource* nodes (or users and resources for short). Resource nodes provide services utilizing their resources such as shared files, memory, or processors to users, while the users submit their jobs, e.g., downloading some files, to request for computation or usage of some types of resources. We study the *resource scheduling/selection* process for P2P applications, which can be further divided into two problems: (1) How to guarantee that users schedule their jobs to the most reliable resource nodes, especially when there exist some selfish and malicious resource nodes in the systems; (2) How to motivate the resource nodes to provide high-quality and reliable resources to the users.

The first problem has attracted a lot of research, most of which are based on reputation and trust for resource scheduling. Selcuk, et al. defined several types of malicious resource nodes in (39) and proposed a reputation-based trust management scheme for P2P networks. Song and Hwang (40) first applied Fuzzy Logic for computing trust values in order to achieve security assurance and resource optimization in Grid Computing systems. Azzedin, et al. (5) suggested that the trust values decay with time and trust relationships be based on a weighted combination of the direct relationships

between domains as well as on the global reputation of the domain. Kamvar, et al. proposed the EigenTrust algorithm (24) for reputation management in P2P networks. One problem in these papers is that they did not clearly differentiate reputation from trust. Reputation and trust are generally confused either as a probability with which the resource is expected to complete a job, or as a mechanism to help users to obtain such a probability. Some papers even referred to *trust rating* as the evaluation result of the transaction between users and resources. Such definition for trust is not accurate in many real-world scenarios.

Less research has been done on the second problem. A few questions are raised, for example, why the resource nodes should provide their resources to the users selflessly, and why they should risk themselves being attacked for providing their resources without any benefit. A similar problem in file-sharing P2P applications such as Gnutella (21) and Kazaa (26) is *free-riding* due to the existence of *free-riders* who only want to download files from others but without sharing their files. Golle, et all. proposed a micro-payment mechanism (20) to encourage file sharing in P2P systems. Ramaswamy and Liu also discussed using utility functions (34) to measure the usefulness of the users. Although our purpose is not to solve free-riding and achieve fairness in P2P systems, we use the same idea to solve the second problem, i.e., users should give rewards or incentives to the resources that provide satisfactory services.

In this thesis, we propose an adaptive reputation-based trust framework for peer-to-peer applications called PeerCredential. In distributed P2P environments such as P2P networks or Grid computing systems, peers (i.e., users) often have to request the services from some unfamiliar peers (i.e., resources) that could be altruistic, selfish, or even malicious. Therefore, it is desirable to design an adaptive trust model to motivate cooperativeness of selfish peers and minimize the risk from malicious peers. We assume the resource nodes are economic or selfish and the users should make payments for using the resources. Economic and selfish nodes aim at maximizing their profit. The

non-malicious resource nodes may compete with each other to gain as much profit as possible. Different from traditional reputation-based trust systems, we categorize the roles of peers into three: users, references, and resources. A peer can take one or more of these three roles at the same time within its application context. We define three basic quantifiable metrics: job satisfactory ratings, reputation, and trust, where reputation is calculated from the user's historical transaction-based job satisfactory ratings and trust is calculated from the user's own view as well as other peers' view (as references) on the reputation of a peer (resource). The metric *trust* can be used to quantify the trustworthiness of peers and provide a trustable way of resource scheduling/selection and access control for peer-to-peer applications. The simulation results show that our framework supports economic resources to achieve long-term high user satisfaction, differentiates malicious nodes from normal ones, and encourages the resources to provide high-quality services.

CHAPTER 2. PREVIOUS WORK

2.1 EigenRep: Reputation Management in P2P Networks (25)

In this paper, Kamvar, et. al. describe a method in a peer-2-peer system to reduce the amount of inauthentic files that are downloaded. The method involves using an algorithm that assigns each peer with a unique global reputation value based upon their history of uploads (25). The reason a system like this is needed is that most P2P systems are anonymous in nature and therefore they are a breeding ground for viruses, trojans, and falsely named files. The EigenRep consists of normalizing the reputation values of each peer so that its value is between 0 and 1. The next thing the algorithm does is aggregate the normalized values by asking other peers their opinions and weighting the opinions based on a value of trust that the requestor has on that peer. The requestor then can make a decision based upon these values. The method for determining reputation in a peer-2-peer described in the paper minimizes the chance that a malicious peer will skew the decision in their favor. By allowing each peer to maintain trust and reputation values of each of their peers, they avoid having a centralized system do all of the work.

2.2 The EigenTrust Algorithm for Reputation Management in P2P Networks (24)

Kamvar, et.al. describe an algorithm for maintaining and receiving reputation values from peers in both a distributed and non-distributed manner. The algorithm collects reputation values from other peers and their peers, etc. and then normalizes the end result in which the user is then able to make a decision regarding which resource to use. This system also relies on a set of pre-defined trusted peers in order for the system to be most effective. The problem with using pre-defined trusted peers is that if one were to turn malicious or get hacked, it would have a negative effect on the entire system similar to that of a house with a crumbling foundation. However, if the pre-defined peers remain untainted, the system is very effective at eliminating the use of malicious peers. Another issue that this algorithm faces is the amount of iterations required to receive trust values becomes increasingly lengthy as the size of the entire system increases reducing its scalability.

2.3 Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behavior (10)

In this paper, Dellarocas primarily focuses on the trading community and preventing maligned users from intentionally giving unfair or discriminatory ratings regardless of the transaction experience. Though this paper involves human intervention when rating, the mechanisms for providing unbiased references would be highly useful in the distributed computing world. In order to solve the problem of unfair ratings, the author proposes two mechanisms which could solve this quandry. One mechanism is using controlled anonymity during transactions to limit unfair low and negative ratings. The other mechanism is using cluster filtering to help eliminate most of the unfair high or positive

ratings. The research done by the author identified the different scenarios that take place in the trading world and have applied two mechanisms to process the results into a more legitimate rating. Furthermore, the author found that these mechanisms are flexible enough that they work in a variety of settings.

2.4 Cooperative Peer Groups in NICE (27)

Lee, et. al. go into detail on how to implement a non-centralized trust system in to the distributed environment, NICE or NICE is the Internet Cooperative Environment (30). NICE is a platform for implementing cooperative applications for the internet (27). Their focus is on a creating a solution that can identify cooperative and non-cooperative users and creating cooperative groups based on these identifications and applying it as a distributed system. The authors were able to achieve their goals by creating an algorithm that selects a trust path based on whether it is the strongest path or using a weighted sum of strongest disjoint paths (27). In the end, their solution was shown to be effective against malicious users and various attacks.

2.5 Grid Resource Management, Scheduling and Computational Economy (7)

Buyya, et. al. takes in to account the issue of resource management and how to schedule jobs based on an economical concept. In this case, they describe a computational Grid that is made up of several different organizations that use their own management scheduling software. It is suggested that there is an actual cost in being able to access these various organizations for the right to use the resources located there. The problem is that there is no way to determine if that particular organization can meet the other requirements of the job to be submitted other than performance. The solution that they

came up with was an economic based approach for Grid computing. By providing an ability for the client to specify specific parameters and a maximum cost to execute the job, a barter of sorts can take place and the job is sent to the resource that meets the minimum requirements at the lowest cost.

2.6 A Community Authorization Service for Group Collaboration (33)

Laura Pearlman, Von Welch, Ian Foster, Carl Kesselman, and Steven Tuecke discuss the addition of new server to each Grid organization, calling it a community authorization service or CAS for short. The function of the CAS is to provide a third party authorization service that would manage that particular Grid organization's resources. The CAS does this by maintaining a table of resources that the organization has. It maintains a policy that then determines what resources a user may be able to access through various permissions listed in the policy. An advantage to using the CAS system is that local and remote hosts only need to trust the relationships to CAS and not to each other, making this system very scalable. The downside to doing this is that the CAS will be doing all of the resource management and may create a bottleneck at the CAS itself.

2.7 An Online Credential Repository for the Grid: MyProxy (31)

Jason Novotny, Steven Tuecke, and Von Welch discuss the use of Grid portals, which allow access to Grid applications through a web site. The problem they discovered was that the security measures usually used in Grid applications were no longer available through the use of a web server. The authors then designed an online credentials

repository called MyProxy (31) to bridge the incompatibility gap between web and grid security. The requirements for creating such a device include: the ability to use any web browser available to access the Grid portal, to be able to use Grid resources from any location that their credentials would not be readily available to them, and that the Grid user would still be able to perform the duties that their credentials provide them. This is definitely a feature that a lot of Grid users would like to have. Since many Grid users would be scientists and may have to travel frequently, the ability to access the Grid from wherever they may be would allow these scientists to continue to perform their experiments away from the lab.

2.8 A Reputation-Based Trust Management System for P2P Networks (39)

Selcuk, et al. designed a protocol for Peer-2-Peer systems in which a peer that is looking to download a file uses reputation from itself and others to help determine which host to download from. Meanwhile, the authors define four types of malicious resources such as Naive, Hypocritical, Collaborative, and PsuedoSpoofing. The authors compare groups of peers that offer different versions of the same file first and then selects a node that is a subset of the selected group to download the file from. The protocol is too extreme in the evaluation as it appears there is only one criteria for rating a peer and that is if the file is correct or not. However, other criteria.could be examined such as length of time to download. In addition, when a peer is selecting a group to download a file from, it is possible that there may exist an overwhelming majority of positively rated peers in one group only to be tainted by a single peer with a positive distrust rating.

2.9 Evolving and Managing trust in Grid Computing Systems (5)

Azzedin and Maheswaran describe a trust model that can be used for Grid computing. In their model, they determine trust levels of domains within the Grid where domains are groups of resources. Trust levels are determined based on the context of service and the declarations of other domains regarding the targeted group. Unfortunately, calculating trust for a group as a whole, malicious nodes could otherwise prevent perfectly good resources from be selected. However they do try to mitigate against this problem by using member weights as part of the domains evaluation, but systems that have been part of the domain for a longer amount of time could undermine the weighting system by purposely behaving until it had established itself in the domain.

2.10 Reputation based Grid Resource Selection (4)

Alunkal, et. al., convert the EigenTrust model for P2P network to one that understands the Grid framework. A reliability trust is added for generalizing the status of an institution while an entity within the institution is also provided with a trust level. The user uses a heirachical decision process in order to select the best resource using the converted Eigentrust as a model. This system is highly scalable for the Grid due to heirchical format, which keeps the number of needed iterations down. However, this process lacks motivation by any of the refering institutions to process queries correctly.

2.11 A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks (9)

Damiani, et. al., propose a system for peer-to-peer systems in which servents ask for a vote for a proposed resource. They select a subset of votes revceived and then

verify that they originated from the original servent to prevent any shill voting. The problem with using this algorithm to determine the best resource to use is that the other peers or servents have no incentive to provide accurate information or even respond to a query on any given resource, especially ones that perform well in order to prevent overloading their favorite resource. The proposed system does evaluate how votes are cast and reputation is also built for other servents which gets reset if they were to change thier ID to discourage pseudospoofing.

2.12 Free Riding: A New Challenge to Peer-to-Peer File Sharing Systems (34)

Ramaswamy and Liu discuss the free rider problem that infests many peer-to-peer systems. They proposed an incentive system that is aimed at keeping peers on the network to share their files which is based on three characteristics: the amount of files shared, the amount of data the peer has shared, and the popularity of the shared file. Peers are thus rewarded based on these metrics. The authors however do not address the spoofing that may occur when a peer wishes to gain access to the system. It would be possible for the peer to create many files with the name of a popular share in order to receive the reward which is additional download capability. Implementing file hashes would solve one problem, but does not prevent the peer from putting up a significant amount of large sized files in order to enhance its ability to download from the system.

2.13 Incentives for Sharing in Peer-to-Peer Networks (20)

Golle, et. al. proposed using game theory in addressing the free rider problem in peer-to-peer applications. Their goal was to provide an incentive model to peers in such a way that it balances the amount of downloading and uploading they do. To do this,

they introduce a micro-payment system in which peers pay to download and are paid when uploaded from. It was also stated the users would get annoyed from having to pay each and every time a few cents for a download, so they suggest selling blocks of credits to use with downloading. It appears there is no recourse for a peer to correct a transaction in which a bogus file may have been downloaded other than not use that resource again. The only form of motivation to peers to prevent uploading bogus files is that they only receive download credits.

2.14 Internet Auction Sites

Auction sites on the internet have become more popular over the last few years as they attract consumers and businesses alike due to ease of use and the potential deals that can be made. Auction sites are very comparable to a distributed system in that you have a group of buyers (the users) and a group of sellers (the resources). The buyers determine which seller to choose from based on any criteria that is available to them. Usually, the information available consists of a price and a rating. Once a buyer has selected a seller, the transaction is processed. The transaction is complete when both parties have rated one another. eBay (11) and Overstock.com Auctions (32) are two businesses that support a rating system.

One of the most popular web auction sites, eBay, uses a simple rating system for both its buyers and sellers. Customers of eBay are allowed to rate their transaction partner with one of three ratings, positive, neutral, and negative. In addition, both buyers and sellers can make comments on the rating they gave to the other party as well as commenting on any ratings they have received themselves which is useful to explain why they may have received a bad score. All ratings are available to the general public although eBay does offer a service where they can make a user's rating private. The problem with this rating system is that buyers and sellers don't have to leave their

rating right away. In addition, the rating system is misused as a type of blackmail where the seller will not leave positive feedback until they receive positive feedback. Another problem is that of rating manipulation where a seller or buyer uses inexpensive items to build up their rating and then turn around and sell or buy a big ticket item without the intention to actually follow through with the sale. These problems make the rating system inaccurate at best.

Overstock.com Auctions uses a slightly different rating system that allows the buyers and sellers to use more precision in how they rate their transaction partner by allow five levels rather than three. However, Overstock Auctions has the same pitfalls that eBay has and relies on their users to be fair and honest.

CHAPTER 3. PROBLEM STATEMENT

In this section, we discuss the system and threat model. Then, we present our problem definition and four primary goals for our adaptive reputation-based trust framework for P2P applications.

3.1 System Model

In P2P applications, the roles of peers can be categorized as *resource* nodes and *user* nodes. Resource nodes provide services utilizing their resources such as shared files, memory and processors for users. In general, we say users would like to use resources by submitting jobs, and resources provide services by processing these jobs. We model the systems using a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. Each vertex represents a node in the systems. Moreover, we have $V = V_u \cup V_{rs}$, where V_u is the set of users and V_{rs} is the set of resource nodes. We also define $E = \{< x, y > | x \in V_u, y \in V_{rs}\}$, where the edge $< x, y >$ denotes the interaction or relationship between user x and resource y. Compared to the traditional distributed systems, one property of P2P systems is: The roles of resources and users are not static. Physically, a particular node can be either a user or a resource, or even both at the same time, i.e., the node can provide its resource to others and simultaneously submit its own job to request the resources provided by others.

To motivate resources to provide high-quality services to users, we assume users would pay for the jobs they submitted. Let us consider such a scenario: Before each

user submits its jobs, it queries the system for available resources that can fulfil its jobs. Receiving the query, those resources capable and willing to process the jobs would respond to the user with the desired payment (or price) for processing the jobs. Ideally, users and resources are economic. Thus, an economic user would select one or multiple best resources, e.g., those offering the lowest price or most reliable ones (We will discuss how to select the best ones later.) to process its jobs. Meanwhile, the economic resources would compete with each other by offering a price to win the competition in order to maximize the profit. Note: Always offering the lowest or highest price cannot guarantee the maximum profit.

3.2 Threat Model

We assume users pay before using resources, hence, there is no way for users to repudiate. However, there do exist some selfish or malicious resources that refuse to provide satisfactory services or even attach the user through malicious resources, e.g., offering a shared file that contains malicious code after receiving the users' payments. Therefore, we must identify these possible threats.

- **Honest Resource**: An honest resource is not a malicious one in itself, but it follows a strategy that must be addressed. By always providing the lowest possible price and performing jobs to a satisfactory level, the honest resource is attempting to undercut the competition. Its goal is to be a monopoly to the distributed system equivalent to WalMart in the retail trade. This would be very dangerous because it can cause severe damages to the system if it becomes malicious. We denote the set of honest resources by V_{rs_H}.

- **Naive Resource**: A naive resource always offers the lowest price to make sure it is selected by users, however, it would never really provide the services required by the users. Its goal is to disturb the system as much as possible by always providing

false information to inquiring users. Moreover, a naive resource may also choose to provide malicious services, instead of refusing service. V_{rs_N} denotes the set of naive resources.

- **Selfish Resource**: A selfish resource tries to maximize its profit by refusing to provide services to users after receiving their payments. However, to attract more users, a selfish resource may act as an economic one most of the time by providing satisfactory services to the users. It is not a malicious one. We use V_{rs_Sf} to denote the set of selfish resources.

- **Random and Structured Hypocritical Resource**: Hypocritical resource is malicious one, but its behavior is similar to a selfish one because they both provide satisfactory services to users a majority of the time. The differences between them are: (1) Hypocritical resources may provide malicious services, but selfish ones never, except for only refusing to serve the users as requested. (2) They have different purposes by providing satisfactory services to the users. A selfish resource's goal is to maximize its profit, but a hypocritical one only wants to deploy their malicious services by attracting more users to use them.

  The random hypocritical resource is associated with a probability to determine when to act maliciously. For example, if the probability is 20%, it would randomly be malicious in 20% of its lifetime. On the contrary, a structured hypocritical resource does not make decisions to be malicious based on chance. It follows a set pattern to act maliciously. For example, it may be malicious every time after providing four satisfactory services. Hence, it would also be malicious in 20% of its lifetime. We use V_{rs_HyR} and V_{rs_HyS} to denote the set of random and structured hypocritical resources.

- **Collaborative Resource**: Resource nodes and user nodes may be collaborative

and form a special group. They act differently depending on whether the requesting peers are in the same group. For example, a collaborative resource will provide satisfactory services to a user if they are from the same group. Otherwise, it may refuse to provide services or even provide malicious services to the user since the user is outside its group. We further categorize collaborative resources into different types based on three aspects such as Resource Availability, Resource Price and Resource Performance. These aspects specify whether the resource is available to the outsider, what price they offer to the outsider, and what quality of service they provide to the outsider. Specifically, they can say they are available or unavailable to outsider; they can offer the lowest price, the normal price (as to their peers), or a higher price (than the normal one) to outsider; they can refuse to provide their services, or provide the normal or malicious services. We use V_{rs_C} to denote the set of collaborative resources and list all possible options for each aspect in TABLE I. The combinations of these options represent the various types of collaborative resources. Note: Not all combinations are valid. For example, if Resource Availability is unavailable, there is no choice for the other two aspects except for "N/A".

- **PseudoSpoofing Resource**: PseudoSpoofing resources are those nodes who change their identities frequently in case they are recognized by users as being malicious. There are several ways to contain this threate. One solution would be to charge every new node that enters the P2P system in order to make it expensive to repeatedly rejoin. Another method would involve the newcomer to pay other nodes to download from it so that it may grow its reputation. However, since this type of threate can be easily mitigated by any of the above solutions, we will not discuss it further in our paper.

Table 3.1 Various Options in Three Aspects for Different Types of Collaborative Resources

Resource Availability	**Resource Price**	**Resource Performance**
Available or unavailable	Lowest, Normal or Higher	Refusing, Normal or Malicious

Let V_E to denote the economic (or normal) resource. We can further divide the set of resource nodes into various types such that we get:

$$V_{rs} = V_{rs\_E} \cup V_{rs\_H} \cup V_{rs\_N} \cup V_{rs\_Sf} \cup V_{rs\_HyR} \cup V_{rs\_HyS} \cup V_{rs\_C}. \tag{3.1}$$

Note: Although it is possible that there may exist malicious users, e.g., those who submit jobs containing malicious code to crash the resources, this is beyond the scope of this thesis.

3.3 Problem Definition

Considering that P2P systems consist of users and various types of resources, we study the resource management and scheduling problem, i.e., how to efficiently motivate resources to provide satisfactory services to users and how to guide users to select the most reliable resources to fulfill their jobs in the cases there exist even malicious resources. We expect to design a scheme for distributed P2P applications to achieve the following goals:

1. Motivate the economic and non-malicious resources to provide the best services as possible.

2. Reduce the impact that malicious resources have on the system as a whole.

3. Provide a facility for an economic user access the best resources possible.

4. Allow per usage contracts for greater flexibility compared to long term contracts.

CHAPTER 4. A FRAMEWORK FOR SUPPORTING REPUTATION-BASED TRUST FOR P2P APPLICATIONS

4.1 Overview

In our framework, we introduce three roles of peers: *user*, *resource* and *reference*. A reference is a node from whom a user can ask its opinion about a resource. It is needed because a user may not have any experience with a resource in order to evaluate it. After each job is finished, a user gives a *job rating*, which is a value between zero and one to evaluate the quality the job processed by the resource. Based on a sequence of historical job ratings, the user can calculate the *reputation* for each resource to measure the overall quality of services that have been provided. Moreover, combining the job ratings provided by the references, the user calculates the *trust* value for each resource, which indicates the probability that the user can expect the resource to provide a satisfactory service for its next job. Considering both the price and trust of resources, the user can calculate and compare the *risk assessments* to select the most appropriate (reliable) resource to fulfill its job. Meanwhile, all economic resources compete with each other to try to win the user's selection competition in order to maximize their profit. We design a number of strategies for resources that determine the price they should offer to win the competitions in a long term.

4.2 Job Rating

A job contains the requirements for some resources. It may involve the types of resources needed, the deadline to finish the job, whether it needs encryption, or any other requirements. A transaction between a user and a resource is referred as the processing of a job. After receiving the result of the job or timeout, the user evaluates the result of the job. A *job rating* of a user to a resource is a metric used by the user to evaluate the quality of job having been processed. In our approach, the job rating is represented by a value between zero and one. (Note: Job rating is named as *trust rating* in some papers, and some use this rating as a binary value, either zero or one.)

The job rating measures the quality of service the resource provided in a particular transaction. It may depend on various metrics such as how fast the job is finished, whether the resource provides security or privacy mechanisms for processing the job, and so on. For example, the user can set a soft deadline and a hard deadline for each job, while the soft one is ahead of the hard one. If the resource finishes the job before the soft deadline, it is scored with the highest job rating. Otherwise, the rating would go down from one to zero until the hard deadline is missed. The two deadlines would be encapsulated in the query when the user ask for the availability of resources. A resource responds "unavailable" if it figures out it cannot finish the job before the deadlines. The job rating $b_{rs}(x, y)$ of user x on resource y for some transaction can be computed as follows:

$$b_{rs}(x, y) = \frac{\sum_{i=1}^{n} w_{i_x} \cdot q_i(x, y)}{\sum_{i=1}^{n} w_{i_x}}, \tag{4.1}$$

where $q_i \in (0, 1]$ is the score or the rating of one of n metrics used by the user to evaluate the quality of service, and $w_{i_x} \in (0, 1]$ is the weight of the corresponding metric. Typically, $\sum_{i=1}^{n} w_{i_x} = 1$.

4.3 Reputation

The *reputation* of a user on a resource is a metric measuring the overall quality of previous transactions between the resource and the user. In our framework, it is a value between zero and one, calculated from the job ratings recorded by the user. In other papers, it is called *local trust*, which we believe confuses the trust and the reputation and is not accurate. Assume each user stores at most n job ratings of previous transactions for each resource, the reputation $r_{rs}(x, y)$ of user x for resource y is:

$$r_{rs}(x, y) = \frac{\sum_{i=1}^{n} \alpha^i \cdot b_{rs,i}(x, y)}{\sum_{i=1}^{n} \alpha^i}, \tag{4.2}$$

where $b_{rs,i}(x, y)$ denotes the i-th job rating and the first rating corresponds to the most recent job. $\alpha \in (0, 1]$ is a decay factor, which indicates how important the most recent transaction to the reputation. Note: Each user may set the different value of α for itself.

4.4 Reference

Before submitting its jobs, a user needs to evaluate the resources to select the most appropriate one for processing the jobs. Since the user may not have any transaction or sufficient number of transactions with the resource, the user would ask the opinions of other nodes, called *references*, to gain an overall evaluation of the resource. Let V_{rf} denote the set of references. Thus, we add these new type of nodes to our system model so that we have $G = (V, E)$, where $V = V_u \cup V_{rs} \cup V_{rf}$. Physically, each node can be a user, resource, or reference, but we assume a resource node can not be a reference to itself. More strictly, any resource cannot be reference to the same type of resources. Similar to resources, the user should also pay the references for buying their information, and the references would compete with each other to sell their information to the user.

4.5 Trust

In our framework, the opinion or information provided by a reference is just the job ratings that the reference recorded for the resource. Thus, the user can use these job ratings of references to calculate a *trust* value of the resource. We define *trust* as the probability with which the user expects the resource to finish its next job. When a user x receives the job ratings for a particular resource y from n references z_i $(i = 1, \cdots, n)$, and if all nodes use the same decay factor α, then x can compute the reputation $r_{rs}(z_i, y)$ of each reference z_i for resource y. Thus, the trust $tr(x, y)$ of user x to resource y is:

$$tr(x,y) = \frac{r_{rs}(x,y) + \sum_{i=1}^{n} r_{rf}(x,z_i) \cdot r_{rs}(z_i,y)}{1 + \sum_{i=1}^{n} r_{rf}(x,z_i)}, \tag{4.3}$$

where $r_{rf}(x, z_i)$ denotes the reputation of user x to reference z_i. In (4.3), we weight the reputation $r_{rs}(z_i, y)$ by $r_{rf}(x, z_i)$. This illustrates how reliable the user believes the information from each reference z_i.

4.6 Risk Assessment

After computing the trust of each resource and receiving the price offered by each resource, the user can begin to select the most appropriate resource to fulfill its jobs. Generally, the user would like to choose a high-quality resource that also offers an acceptable price. (Certainly, the user can have other metrics to determine the most appropriate one.). In our framework, we define *risk assessment* as a metric used by the user to select the most appropriate resource. If only considering the price and the quality of service of resource, the risk assessment $v(x, y)$ computed by a user x for a resource y is:

$$v(x,y) = pr(y) \cdot (1 - tr(x,y)), \tag{4.4}$$

where $pr(y)$ denotes the price offered by resource y. The user would select the resource who has the lowest value of *risk assessment*. A risk assessment brings a balance between the quality of service and the price offered by a resource.

4.7 Reference's Adaptive Adjustment on its Reputation Values

We should notice that references can also be malicious. They may provide false information and mislead the user to choose a poor or malicious resource. Similar to the classification of resources, we define various types of references:

$$V_{rf} = V_{rf\_E} \cup V_{rf\_N} \cup V_{rf\_HyR} \cup V_{rf\_HyS} \cup V_{rf\_C}, \tag{4.5}$$

where V_{rf_E} denotes the normal or economic references and $V_{rf\_N}, V_{rf\_HyR}, V_{rf\_HyS}$ and V_{rf_C} denote other malicious references, whose definitions are similar to those of malicious resources.

Therefore, the user needs a way to evaluate references to mitigate against the malicious ones. Since the information provided by the reference helps the user predict the quality of service for the resource, the closer the information to the real result, the more reliable the reference would be. For a particular resource y and a transaction, we define $b_{rf}(x, z)$ as the rating of a user x to a reference z:

$$b_{rf}(x, z) = 1 - |b_{rs}(x, y) - r_{rs}(z, y)|, \tag{4.6}$$

where $b_{rs}(x, y)$ denotes the job rating of this transaction between x and y, and $r_{rs}(z, y)$ is the reputation of reference z for resource y computed by user x from the information collected from z (If x and z have a different decay factor α, x should use $\overline{r}(z, y)$) instead of $r(z, y)$.). $b_{rf}(x, z)$ will be stored by user x and used to update $r_{rf}(x, z)$ by the way similar to (4.2).

4.8 Strategies for Resources to Achieve Their Profit Goal

A resource receives the payments from a user for processing the user's jobs. Except for malicious resources, a normal one must be economic, i.e., the purpose of an economic

		B		
		1	2	3
A	1	(1/2, 1/2)	(1, 0)	(1, 0)
	2	(0, 1)	(1, 1)	(2, 0)
	3	(0, 1)	(0, 2)	(3/2, 3/2)

(a) Initial Payoff Matrix for the Game between Resource A and B

		B		
		1	2	3
A	1	(1/2, 1/2)	(1, 0)	(1, 0)
	2	(0, 1)	(1, 1)	(2, 0)
	3	~~(0, 1)~~	~~(0, 2)~~	~~(3/2, 3/2)~~

(b) The Resultant Matrix after A Removes the Strictly Dominated Strategy 3

		B		
		1	2	3
A	1	(1/2, 1/2)	(1, 0)	(1, 0)
	2	(0, 1)	(1, 1)	(2, 0)
	3	~~(0, 1)~~	~~(0, 2)~~	~~(3/2, 3/2)~~

(c) The Resultant Matrix after B Removes the Strictly Dominated Strategy 3

Figure 4.1 Find Nash Equilibrium based on Iterated Strict Dominance

resource is to maximize its profit. In the following sections, we will discuss a number of strategies for resources to fulfill this purpose.

4.8.1 Binary and Exponential Strategies

Suppose a resource can offer a price from $\$1, \cdots \$n$. Here, we assume the price is the net income where the cost has been deduced.

For simplicity, we first consider such a scenario: There are two resources denoted as A and B. They happen to have the same reputations. Hence, the price they offered uniquely determine who is winner. If they offer the same price, the user may randomly select any one. In this case, their expected profit is the half of the price. Let us assume the maximum price is \$3.

We can regard the competition between these two resources as a simple game. The payoff matrix of resource A and B is shown in Fig. 4.1(a), where each resource has three strategies that correspond to the price they can offer. The first element of each entry is the expected profit of resource A, while the second one is that of resource B. Based on Game Theory, we can solve the *Nash Equilibrium* of this game by finding the *iterated strict dominance* (18): For resource A, its profit vector for strategy 3 is $(0, 0, \frac{3}{2})$, while that of strategy 2 is $(0, 1, 2)$. Since each element of strategy 3's vector is less than that

of strategy 2's, strategy 3 is strictly dominated. Hence, resource A would never select strategy 3. Fig. 4.1(b) shows the resultant matrix by removing strategy 3 of resource A. We assume resource B also knows this. Now for resource B, strategy 3 becomes strictly dominated by strategy 2 since the corresponding profit vector $(0,0) < (0,1)$. Hence, B would never choose strategy 3, and the resultant matrix is shown in Fig. 4.1(c). Repeatedly, the final strategy for A and B would be strategy 1. This process of finding iterated strict dominance can be applied to the case when n is set to be an arbitrary value. Therefore, our conclusion is: If playing one time of this game, the best strategy of all resources would be offering the lowest price.

Now consider a more complicated scenario: There are a total of m resources in competition. Each time there are a total of k jobs generated in the system, and each resource node can only process one job per time. If $k \geq m$, a clever resource can select a strategy by always offering the highest price. Clearly, the users would first select those resources offering the lower price. However, after other resources have been selected, the clever resource would be the only available one. Since there are sufficient number of un-submitted jobs, this resource will be definitely selected even offering the highest price. Thus, this strategy can maximize the resource's profit. If $k < m$, according to the discussion of the simple game, all resources have to offer the lowest price in order to win the selection. We call this strategy *Binary Strategy* in which the resources would offer either the lowest or the highest price depending on current load of the system, i.e., the current number of jobs produced.

Our system is dynamic because each user will generate its jobs independently and the length of each job is not identical. Hence, it is hard for resources to measure the current load of system accurately. The number of available resources cannot be accurately estimated either. Considering this dynamic system model and long-term competition between resources, *Binary Strategy* may not be the best. Therefore, we modify the *Binary Strategy* to be a *Exponential Strategy*. In *Exponential Strategy*, each resource

tries to measure the current load and compare it with the capacity of system, i.e., the total number of jobs all resources can process per time unit. The price a resource offers is an exponential function of the difference between the current load and the system capacity. The idea is that if the current load is sufficiently larger or less than the system capacity, the resource will be more confident to ignore the error in measurement and decide to choose the highest or lowest price correspondingly. The following equation shows how to compute the price in *Exponential Strategy.*

$$pr = pr_{max} \cdot \frac{1}{1 + e^{-k(m-a)}}, \tag{4.7}$$

where pr_{max} is the highest price a resource can offer, m is the measured load, a is the predetermined system capacity, and k is a factor used to adjust the slope of the function. The greater the value of k, the closer the function to be a binary one.

4.8.2 Historical Strategy

Binary Strategy and *Exponential Strategy* are only concerned about the difference between the current load and system capacity. There exists other information, e.g., the previous prices offered by other competitors, which may be useful for a resource to determine their price. The idea is one can estimate the possible next price of its competitors via the previous prices offered by those ones, because when this resource once had transactions with its competitors when it acted as user at that time. Hence, we propose an estimation-based strategy, called *Historical Strategy.*

Assume two resources y_1 and y_2 are competing with each other for the selection by user x. User x would select one resource by finding the lower of the two risk assessments $v(x, y_1) = pr(y_1) \cdot (1 - tr(x, y_1))$ and $v(x, y_2) = pr(y_2) \cdot (1 - tr(x, y_2))$. To win y_2 in the competition, y_1 wants to find $\hat{pr}(y_2)$ the estimated price of y_2, and $\hat{tr}(x, y_2))$ the estimated trust of x to y_2. After that, y_1 can determine an appropriate price $pr(y_1)$ that

can make it to win y_2. Equivalently, y_1 would like to decide $pr(y_1)$

$$\hat{v}(x, y_1) = pr(y_1) \cdot (1 - \hat{tr}(x, y_1)) < \hat{v}(x, y_2) = \hat{pr}(y_2) \cdot (1 - \hat{tr}(x, y_2)), \tag{4.8}$$

where $\hat{tr}(x, y_1)$ is the estimated trust of x to y_1 computed by y_1 itself. Since y_1 knows the quality of every transaction between itself and x, it can easily estimate $\hat{tr}(x, y_1)$. Meanwhile, it can also estimate

$$\hat{tr}(x, y_2) = tr(y_1, y_2). \tag{4.9}$$

The idea behind this estimation is that y_1 assumes y_2 provided similar quality of services to x and itself, hence, its trust to y_2 can be used to estimate x's trust to y_2. We further assume y_1 uses the following equation to estimate the price of y_2 based on the previous prices it received from y_2.

$$\hat{pr}(x, y_2) = \frac{\sum_{i=1}^{n} \beta_i \cdot pr_i(y_1, y_2)}{\sum_{i=1}^{n} \beta_i}, \tag{4.10}$$

where $pr_i(y_1, y_2)$ denote the i-th price of y_2 stored by y_1, and β is a decay factor.

4.8.3 Hybrid Strategies

Historical Strategy is a highly competitive strategy when trying to get selected, however it is unable to capitalize on its reputation when the number of jobs exceed the amount of resources available. This is the result of its pricing strategy in which it still attempts to compete with the best appraised resources. In an attempt to alleviate this problem, we suggest two more strategies, *Historical-Binary Hybrid Strategy* and *Historical-Exponential Hybrid Strategy.*

The *Historical-Binary Hybrid Strategy* is the result of a resource using historical analysis whenever the resource utilization is below full capacity. Once capacity has been achieved, the resource uses the binary strategy of selecting the maximum price allowed by the system. This strategy allows the resource to remain competitive with

other nodes when resource utilization is light but attempts capitalize on its reputation with the maximum price bidding when resource utilization is heavy as it no longer has to worry about being selected with enough jobs to go around.

The *Historical-Exponential Hybrid Strategy* is very similiar to that of the *Historical-Binary Hybrid Strategy* with one exception: instead of bidding with the maximum price when resource utilization is greater than the load threshold, it uses the exponential strategy to make its bid. This enables the *Historical-Exponential Hybrid Strategy* to compete with cheaper resources when the job load is light and compete with the *Exponential Strategy* when the job load is heavy.

CHAPTER 5. PERFORMANCE EVALUATION AND RESULTS

5.1 Evaluation Metrics

To evaluate the performance of our approach, we measure the following metrics:

1. *User Satisfaction*: It is defined as the ratio of the amount of jobs deemed successful by all users over the total amount of jobs run within every equal-length period of time . This will help us determine whether or not malicious resources have any impact on the job scheduling process.

2. *Average Resource Reputation*: It is defined as the average value of the reputations of all users for every resource at a given time instant. This metric is valuable to help users to differentiate various types of malicious resources.

3. *Average Number of Times Resource Selected*: It measures the average number of times any resource of every type has ever been selected. This is another value that helps us determine if users are selecting reliable resources at a greater frequency than malicious resources.

4. *Average Resource Earnings*: It measures the average amount of income any resource of every type has earned. It allows us to identify whether or not incentives are being applied to the appropriate resources. In addition, it helps us to identify what the most profitable strategy is for an normal resource.

Table 5.1 Basic Configuration

Node Types	Number
Historical Strategy Resource / Reference	3 / 21
Exponential Strategy Resource / Reference	3 / 21
Binary Strategy Resource / Reference	3 / 21
Honest Resource / Reference	2 / 1
Naive Resource / Reference	2 / 4
Random Hypocritical Resource / Reference	2 / 4
Structured Hypocritical Resource / Reference	2 / 4
Collaborative Resource / Reference	3 / 4

5.2 Simulation Setup

Our simulation consists of 100 nodes including 20 resources and 80 references. All nodes can act as users. For simplicity, we only have one type of job for which resources can process. For every time slot, called a *tick*, each user generates a number of jobs. The interval between the jobs generated by each user satisfies an exponential distribution, whose parameter determines whether it is a normal load or heavy load in our system. The length of each job is a random number drawn from $(0, 10]$ with an average job length of 5 ticks. We assume each resource can process 3 jobs simultaneously. Hence, the whole system can process 20×3 jobs in 5 ticks, which leads to a system capacity 12 jobs per tick. We set a satisfactory threshold 0.75 to measure if the result of a finished job is satisfactory by the user. The quality of each job processing is randomly determined depending on whether the resource is reliable or malicious. For a job processed by a reliable resource, a job rating is a random number drawn from $[0.75, 1]$ with an average rating 0.875, while for a malicious resource, the job rating is always 0.01. The minimum and maximum price a resource can offer are \$.01 and \$100, respectively. In our simulation, the decay factor $\alpha = \beta = 0.9$.

To test our framework, we built a variety of simulation setups to test our approach and to compare the set of non-malicious resource strategies. The first type of setup that we used is called the Malicious Collaborative Group Configuration. In this configuration,

we attempt to spread out the strategic resources evenly while attempting to keep the total number of potentially malicious nodes as a minority. The configuration is very hostile considering the amount of malicious resources allocated, however we expect that the real world application of this system would have considerably less. In this particular configuration, the probability for random hypocritical nodes being malicious is 20%, and the pattern for structured hypocritical nodes is to be malicious every four jobs. The purpose of the collaborative nodes is to provide malicious services to the non-group members. If the user is not within the same collaborative group as the resource and reference, the resource and reference will offer the lowest price and perform maliciously. Otherwise, they act normally by choosing the historical strategy since the user is their group member. This group uses the parameters found in Table 5.2:

Table 5.2 Malicious Collaborative Group

Res. Avail.	**Res. $**	**Res. Perf.**	**Ref. Avail.**	**Ref. $**	**Ref. Perf.**
Allow	Lowest	Malicious	Allow	Lowest	Always False

In our strategy comparison, we first test each type of the hybrid resources against the Malicious Collaborative Configurations that the historical resources already had to endure. We then simulated one on one comparisons of the historical, binary hybrid, and exponential hybrid strategies against the exponential strategy. In these simulations, all types of resources and references were removed with the execption of those being tested. We wanted to observe whether or not historical, binary hybrid, or exponential related resources would be influenced by the cheaper reliable and unreliable nodes. As a result, we have 10 historical, historical-binary, or historical-exponential resources and 10 exponential strategy resources, 40 historical, historical-binary, or historical-exponential references and 40 exponential strategy references.

A simulation is complete when 10000 ticks have been completed. We ran the simulation 5 times for each data set in order to create the graphs shown in Appendix A. This was done in order to eliminate any possible anomalies. For each configuration we

tested, we gathered a data set that was based on a normal load and a heavy load. A normal load in our configuration should on average generate 6 jobs per tick, while the heavy load parameter generates on average 12 jobs per tick.

5.3 Simulation Results

5.3.1 User Satisfaction

Normal and Heavy Load Configuration with Malicious Collaborative Group Results: Fig. A.1(a) and Fig. A.1(b) shows that our approach offers high user satisfaction over both resources and references very quickly. The user satisfactions converge to more than 90% only after 2000 ticks in the normal load configuration and 3000 ticks in the heavy load configuration, which indicates that user nodes are using reliable resources and references before they are forced to use malicious ones or wait for available reliable ones. In our simulations, we discovered that regardless of the job load in the system, user satisfaction remained constant.

5.3.2 Average Resource Reputation

Normal and Heavy Load Configuration with Malicious Collaborative Group Results: Fig. A.2(a) and Fig. A.2(b) illustrates how fast the average reputation of each type of resources converges. In the simulation, this metric is only calculated based on the job ratings of those users that had used a resource. Figs. A.2(a) and A.2(b) indicate our approach can differentiate malicious resources quickly. In our simulation, the job rating for a reliable resource is 0.875 in average, while that for a malicious one is 0.01. Therefore, the average reputation of a binary, historical, exponential strategy and honest resource should converge to around 0.9, while the naive one converges to 0.01. For a hypocritical resource, no matter whether it is random or structured one, it will be malicious with a probability 20%. Hence, it will gain a rating of 0.01 with a probability

20% and a rating of 0.875 with a probability 80%. Its reputation will converge to $0.01 \times 20\% + 0.875 \times 80\% \simeq 0.7$, which is consistent with the simulation results. The case for collaborative resource is a bit more complicated than the others. A collaborative resource would be malicious to non-members and reliable to members. In our simulation configuration, a collaborative resource has 93 non-member nodes who are likely to give a 0.01 rating and 6 member nodes giving a rating of 0.875. Hence, we would expect the collaborative node to eventually converge around $0.01 \times \frac{93}{99} + 0.875 \times \frac{6}{99} \simeq .07$.

5.3.3 Average Number of Times Resource Selected

Normal and Heavy Load Configuration with Malicious Collaborative Group Results: In Figs. A.3(a) and A.3(b), we show how the average number of times a resource selected increases as time passes under a normal and heavy load. It is simple to show that honest resources will be selected most often as they are reliable and always offer the services with the lowest price. The same effect can be seen with the binary nodes as well because there is not enough resource utilization to trip its highest price, therefore, they effectively act as a honest resource. The historical resource is selected considerably less due to the fact it has a slightly higher price than the lowest one, thus, it is the next in line to be selected by users after the honest and binary ones are filled. Both types of hypocritical resources are only 80% as reliable as a historical one, so it is reasonable that they would be selected 20% less than historical one, even thought they follow the same pricing strategy. In the long term, collaborative resources will only be selected by its own members. Therefore, its curve is linear but only 6% as the honest and binary ones are because only 6 users are able to utilize it positively. The exponential ones are only selected after users are able to filter malicious ones such as naive. Due to their higher price, they can only be selected after historical resources are filled and the user has had enough bad experiences with the hypocritical resources.

Historical-Binary in Configuration with Malicious Collaborative Group: Fig. A.5(a)

we see that the average number times that the historical-binary node has been selected has increased compared to the historical resources shown in Fig. A.3(a). In contrast, in Fig. A.5(b), there appears to be a slight decline in the number of time selected as compared to the historical resource. This slight decline could be a result of using the maximum price as a bid when it thinks that the load is high. The problem is that the the other nodes will have a lower price at the 12 job per tick threshold and they will be utilized first at that point. However, neither the positive and negative variances in resource selection are great enough to be significant in this simulation.

Historical-Exponential in Configuration with Malicious Collaborative Group: In Fig. A.6(a), the amount of times that the historical-exponential resource was selected is equivalent to that of the historical resource shown in fig. A.3(a). The reason is most likely due to the fact that the historical-exponential node was operating as a historical node most of the time.

5.3.4 Average Resource Earnings

Normal and Heavy Load Configuration with Malicious Collaborative Group Results: Figs. A.4(a) and A.4(b) show the average income that each type of resource has earned at any time. Interestingly, though the exponential resource is selected the fewest amount of times of all of the reliable resources, it yields the most income of all the nodes. It earns less at the beginning, however once it gains reputation, it could get selected at a higher value that would remain constant if resource utilization does not change. Honest resources have stabilized on a low price, although slightly higher than that of honest and binary, therefore, they receive greater income. Both hypocritical nodes also use the same pricing strategy as that the historical resource does, but are selected less often due to the 20% chance of being malicious. Collaborative resources are basically historical resources to a much smaller audience, therefore, are going to have even less income. Honest and binary income is equivalent to 1/100 of their selection frequency because

they earn $0.01 every time selected.

Historical-Binary in Configuration with Malicious Collaborative Group: Fig. A.7(a) and Fig. A.7(b) show practically the same results as those shown in Fig. A.4(a) and Fig. A.4(b). This is a result of the high price that the binary-historical resource has when the system is heavily utilized. Since its price is greater than any of the other nodes when resources are being heavily used, it will be selected last if at all.

Historical-Exponential in Configuration with Malicious Collaborative Group: Fig. A.8(a) shows that the historical-exponential resource's income average reflects that of the the historical resource income average as the average load for the system is mostly below the threshold required to use the exponential pricing algorithm.

5.3.5 Strategy Comparison Configuration Results

In this configuration, we compare the performance of historical strategy and exponential strategy by measuring the average times a resource is selected and the average income a resource has earned. First we will show how each of the hybrid strategies perform in the configuration with the malicious collaborative group in both normal and heavy load scenarios, and then we eliminate all other node types beside the hybrid and the exponential resource to see if the "cheap" nodes had any effect on the historical or hybrid resources.

5.3.5.1 Average Number of Times Resource Selected

Historical vs. Exponential: In Fig. A.9(a) and Fig. A.9(b), we see that the historical resources are being selected more often than the exponential, due to their low price. Once the historical resources have been fully selected, users have no choice but to select the exponential resources. In the heavy load scenario, it appears that the amount of times that each type of node gets selected eventually converges to about the same amount. The

primary reason that these two types of resources eventually converge is that the amount of jobs being created should eventually meet the expected supply by the resources.

Historical-Binary vs. Exponential: In Fig. A.10(a), the historical-binary resource is seen being selected more often than the exponential as it is primarily acting as a historical resource with resource utilization being down. However the times that the resource utilization does cross the threshold set by the historical-binary resource, leaves it vulnerable to the pricing strategy of the exponential resource, allowing them to be picked first. In the heavy load scenario A.10(b), the above scenario is amplified in such a way that the exponential pricing scheme is actually cheaper to select from. This results in the historical-binary resources being selected less.

Fig. A.6(b) shows a higher selection average for the historical-exponential hybrid than the historical in A.3(b). This is unexpected as the result should have been lower due to higher prices it was bidding. The most probable explanation is that the amount of times that hypocritical and binary resources were selected was lower in Fig. A.6(b). This would have resulted in the historical-exponential nodes taking up the slack when resource utilization was low. A reason that the hypocritical nodes are lower is that they are bit more volitile than the other nodes and will have variances one simulation to the next. The reason the binary nodes had reduced its average selection rate is that the hybrid nodes are priced more competitively at higher loads and therefore will take away from the binary resource's selection average.

Historical-Exponential vs. Exponential: In Fig. A.11(a), the curves match directly with that of the historical vs. exponential in Fig. A.9(a). However in the heavier loaded Fig. A.11(b), the historical-exponential curve is the only one of the three strategies compared against the exponential strategy to not be concave. This is most likely due to its ability to compete with the exponential resource in both normal and heavy load situations.

5.3.5.2 Average Resource Earnings

Historical vs. Exponential: In Fig. A.12(a) and Fig. A.12(b), the exponential resource earns more income even though it is much less selected as the historical one does. Since the historical resource uses a pricing strategy that is dependent on the lowest risk assessment of any resource peer, its price will remain lower than that of exponential one. As a result, any historical resources will get selected before exponential resources. Due to the price differential, once exponential resources are selected, they are able to make a much larger profit than the historical ones are. Therefore, we can conclude that the exponential strategy is better than historical one in maximizing the resource's profit. However, the historical one is also useful to make the malicious ones get less selected. Otherwise, the users have to experience more malicious services.

Historical-Binary vs. Exponential: In Fig. A.13(a) and Fig. A.13(b), these curves are not much different than the previous two figures A.12(a) and A.12(b). However it should be noted that althought the historical-binay resource performed the same in income to the previous configuration involving the historical resources, it did not have to be selected as often to acheive that price.

Fig. A.8(b) shows that the historical-expoenential resource has a much higher income average than what a historical resource (Fig. A.4(b) would have. However it still is not equal to the average income that the expontial node has. In addition, it is utilized much heavier than the exponential resource to even achieve the higher income average which would reduce its net income gain.

Historical-Exponential vs. Exponential: In Fig. A.14(a), this graph is really no different than the other configuration's normal load income graphs as it mostly acts as a historical resource with the load utilization down. Fig A.14(b) was a real surprise, due to the fact that the historical-exponential is looking to beat the exponential resource in total income in the long run. It appears that the "cheap" nodes do have an effect on

the hybrid. However, even though the historical-exponential node is equivalent to the income average of the exponential resource, it is still heavily utilized and would most likely result in smaller net gain. We have to conclude that the exponential resource is the most ideal model for resources to follow if monetary gain is important.

CHAPTER 6. CONCLUSION

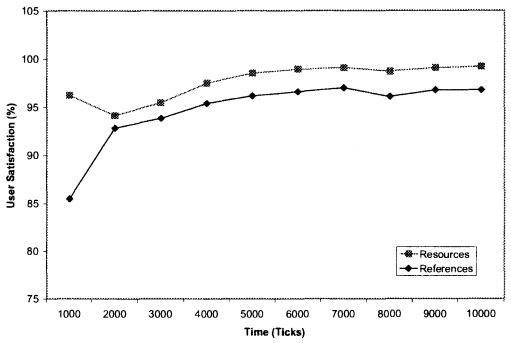
In this paper, we propose an adaptive reputation-based trust framework for peer-to-peer applications. Besides malicious nodes, we assume there exist economic and selfish nodes who compete with each other to maximize their profit. Different from traditional reputation-based trust systems, we categorized the roles of peers into three: users, references, and resources. A peer can take one or more of these three roles at the same time within its application context. We defined four basic quantifiable metrics: job satisfactory ratings, reputation, trust, and appraisal. The metric *trust* can be used to quantify the trustworthiness of peers and provide a trustable way of resource scheduling/selection and access control for peer-to-peer applications. We select an appropriate resource to serve the user request by quantifying and comparing the appraisal of the resources. The simulation results show that our framework supports economic resources to achieve long-term high user satisfaction, differentiates malicious nodes from normal ones, and encourages the resources to provide high-quality services.

CHAPTER 7. FUTURE WORK

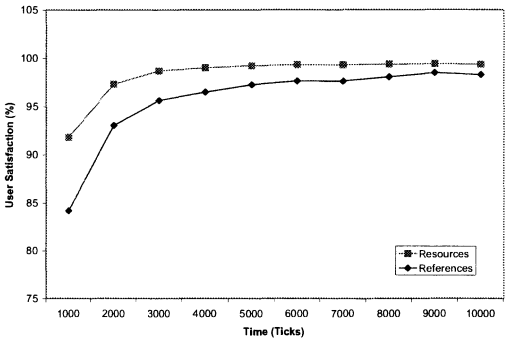
There is much more that can be done to enhance the PeerCredential framework. Historical node strategy could possibly be improved by implementing price thresholds to prevent the bidding price from being too low. Simulations of nodes joining and leaving the P2P system should be done to investigate the effects it has on the system and the inclusion of more than two collaborative groups are needed to provide a more real world like scenario.

APPENDIX A. SIMULATION GRAPHS

In this Appendix, we display all the charts that have been referred to in the thesis.

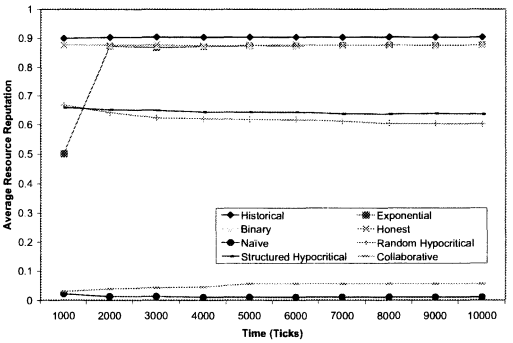


(a) Normal Load

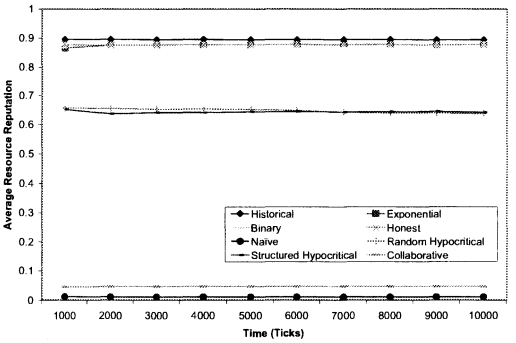


(b) Heavy Load

Figure A.1 User Satisfaction for Configuration with Malicious Collaborative Group

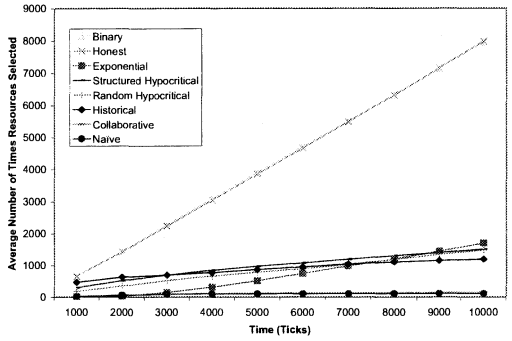


(a) Normal Load

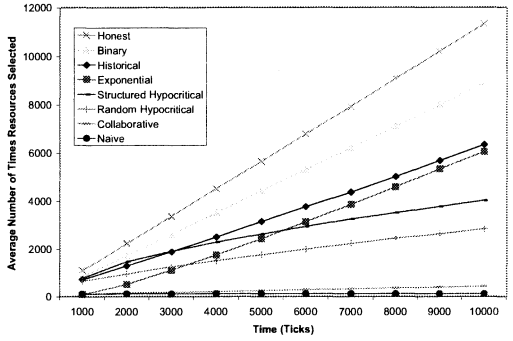


(b) Heavy Load

Figure A.2 Average Resource Reputation for Configuration with Malicious Collaborative Group

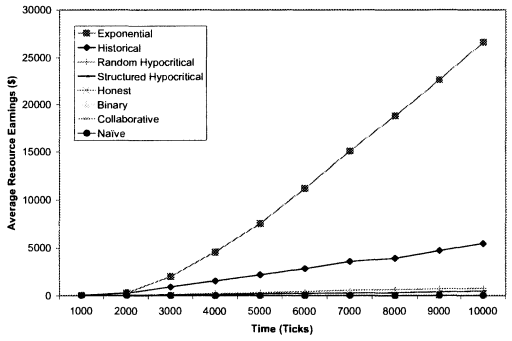


(a) Normal Load

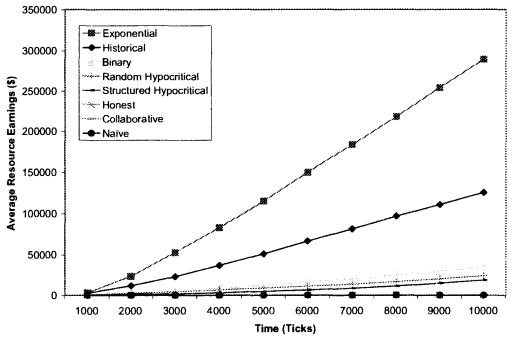


(b) Heavy Load

Figure A.3 Average Number of Times Resource Selected for Configuration with Malicious Collaborative Group

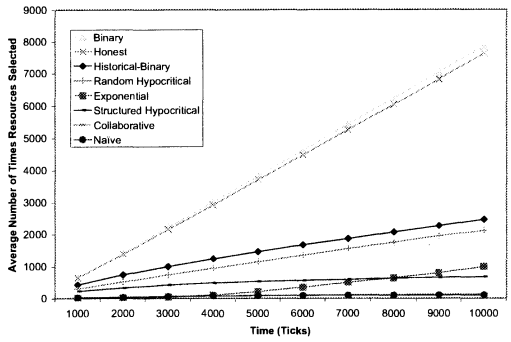


(a) Normal Load

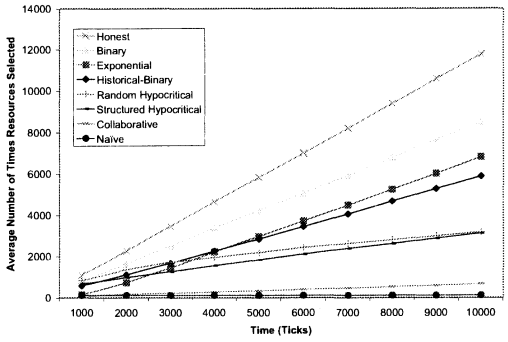


(b) Heavy Load

Figure A.4 Average Resource Earnings for Configuration with Malicious Collaborative Group

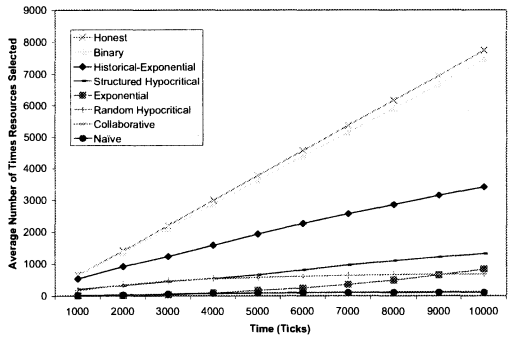


(a) Normal Load

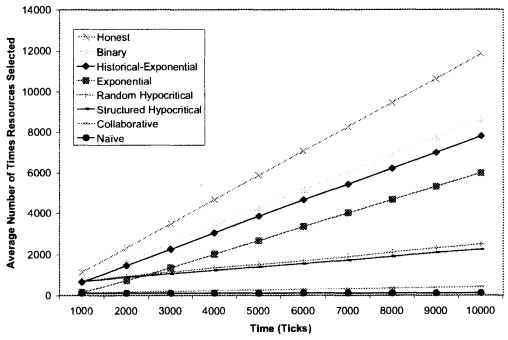


(b) Heavy Load

Figure A.5 Average Number of Times Resource Selected in Historical-Binary for Configuration with Malicious Collaborative Group

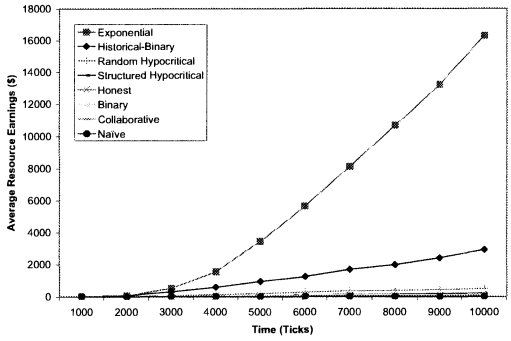


(a) Normal Load

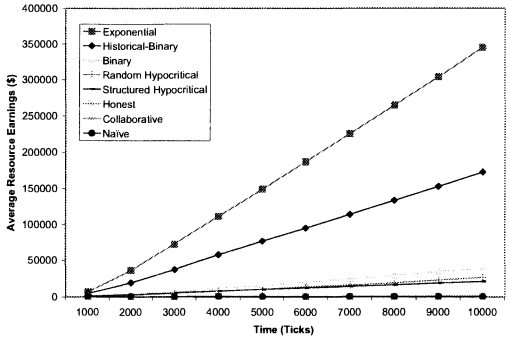


(b) Heavy Load

Figure A.6 Average Number of Times Resource Selected in Historical-Exponential for Configuration with Malicious Collaborative Group

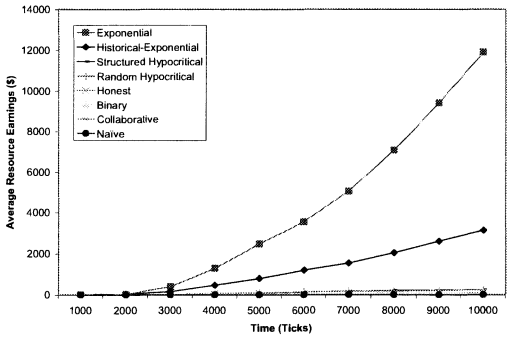


(a) Normal Load

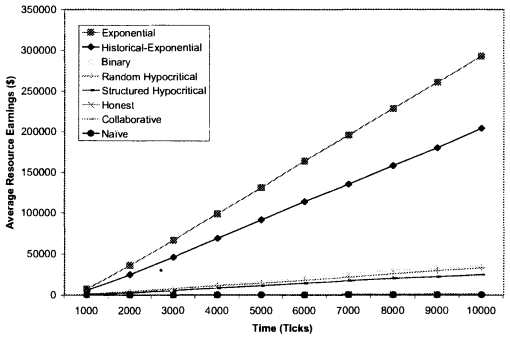


(b) Heavy Load

Figure A.7 Average Resource Earnings in Historical-Binary for Configuration with Malicious Collaborative Group

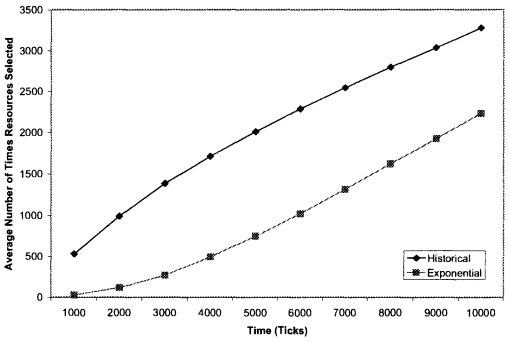


(a) Normal Load

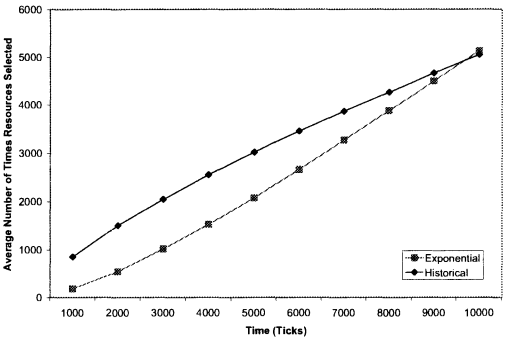


(b) Heavy Load

Figure A.8 Average Resource Earnings in Historical-Exponential for Configuration with Malicious Collaborative Group

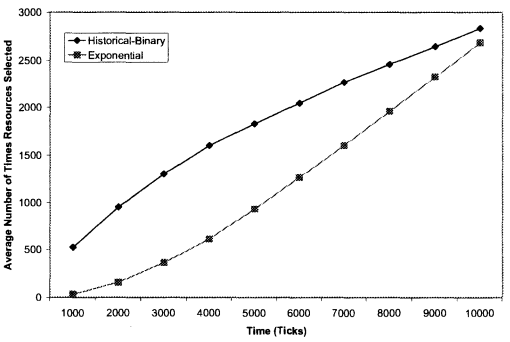


(a) Normal Load

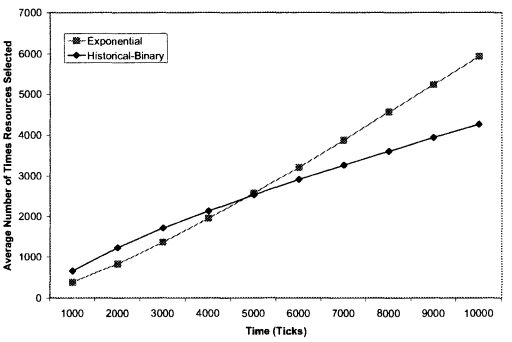


(b) Heavy Load

Figure A.9 Average Number of Times Resource Selected in Historical vs. Exponential

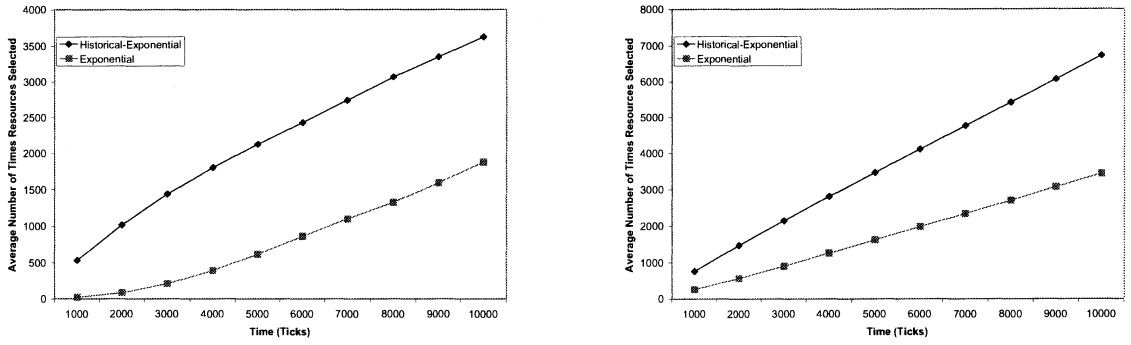


(a) Normal Load



(b) Heavy Load

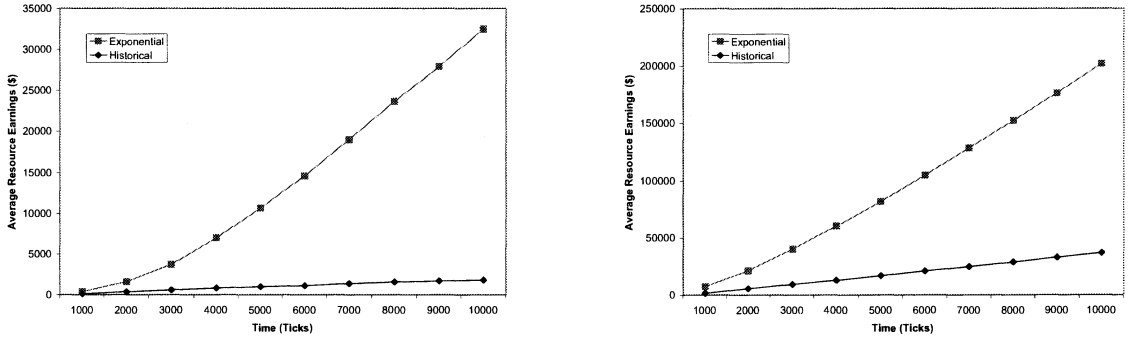
Figure A.10 Average Number of Times Resource Selected in Historical-Binary vs. Exponential



(a) Normal Load

(b) Heavy Load

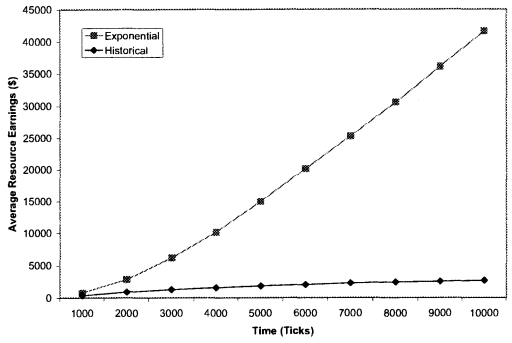
Figure A.11 Average Number of Times Resource Selected in Historical-Exponential vs. Exponential



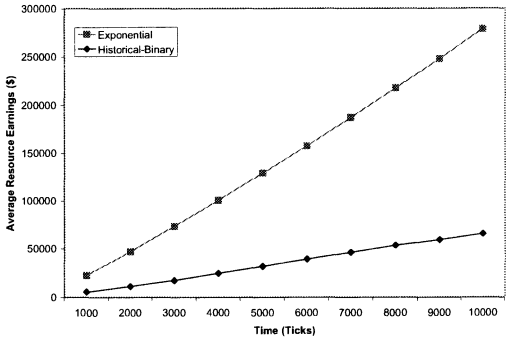
(a) Normal Load

(b) Heavy Load

Figure A.12 Average Resource Earnings in Historical vs. Exponential

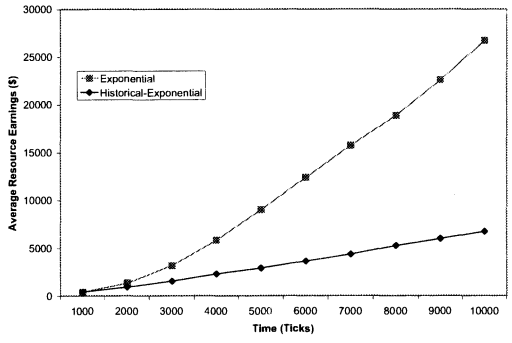


(a) Normal Load

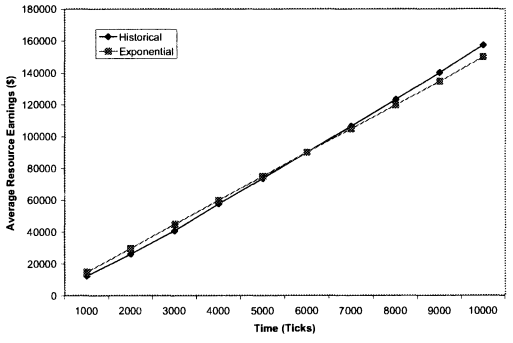


(b) Heavy Load

Figure A.13 Average Resource Earnings in Historical-Binary vs. Exponential



(a) Normal Load



(b) Heavy Load

Figure A.14 Average Resource Earnings in Historical-Exponential vs. Exponential

APPENDIX B. COLLABORATIVE GROUPS

In this Appendix, we list all of the possible collaborative groups that may inhabit a P2P system. Here are a list of definitions that describe each of the actions a collaborative node may make:

- *Allow - Availability*: Allow non-collaborative users to use the resource or reference.
- *Deny - Availability*: Deny access to non-collaborative resources or references .
- *Lowest - $Price*: Bid the minimum amount allowed by the system.
- *Normal - $Price*: Bid using the historical node's algorithm.
- *Higher - $Price*: Bid an amount that is above the result of using the historical node's algorithm.
- *Normal - Performance*: Perform the job or provide information as a historical resource or reference would.
- *Malicious - Performance*: Perform the job as a naive resource would.
- *Always False - Performance*: Always provide false information regarding all references.
- *Always Positive - Performance*: Always provide positive information regarding all resources.

- *Hype - Performance*: Provide negative information regarding non-collaborative resources and provide positive information regarding collaborative resources.

Table B.1 Listing of all Collaborative Groups, Part 1

Res. Avail.	**Res. $**	**Res. Perf.**	**Ref. Avail.**	**Ref. $**	**Ref. Perf.**
Deny	N/A	N/A	Deny	N/A	N/A
Deny	N/A	N/A	Allow	Lowest	Always False
Deny	N/A	N/A	Allow	Normal	Always False
Deny	N/A	N/A	Allow	Higher	Always False
Deny	N/A	N/A	Allow	Lowest	Normal
Deny	N/A	N/A	Allow	Normal	Normal
Deny	N/A	N/A	Allow	Higher	Normal
Deny	N/A	N/A	Allow	Lowest	Always Pos.
Deny	N/A	N/A	Allow	Normal	Always Pos.
Deny	N/A	N/A	Allow	Higher	Always Pos.
Allow	Lowest	Malicious	Allow	Lowest	Always False
Allow	Lowest	Malicious	Allow	Normal	Always False
Allow	Lowest	Malicious	Allow	Higher	Always False
Allow	Lowest	Malicious	Allow	Lowest	Normal
Allow	Lowest	Malicious	Allow	Normal	Normal
Allow	Lowest	Malicious	Allow	Higher	Normal
Allow	Lowest	Malicious	Allow	Lowest	Always Pos.
Allow	Lowest	Malicious	Allow	Normal	Always Pos.
Allow	Lowest	Malicious	Allow	Higher	Always Pos.
Allow	Lowest	Malicious	Allow	Lowest	Hype
Allow	Lowest	Malicious	Allow	Normal	Hype
Allow	Lowest	Malicious	Allow	Higher	Hype
Allow	Lowest	Normal	Allow	Lowest	Always False
Allow	Lowest	Normal	Allow	Normal	Always False
Allow	Lowest	Normal	Allow	Higher	Always False
Allow	Lowest	Normal	Allow	Lowest	Normal
Allow	Lowest	Normal	Allow	Normal	Normal
Allow	Lowest	Normal	Allow	Higher	Normal
Allow	Lowest	Normal	Allow	Lowest	Always Pos.
Allow	Lowest	Normal	Allow	Normal	Always Pos.
Allow	Lowest	Normal	Allow	Higher	Always Pos.
Allow	Lowest	Normal	Allow	Lowest	Hype
Allow	Lowest	Normal	Allow	Normal	Hype
Allow	Lowest	Normal	Allow	Higher	Hype
Allow	Normal	Malicious	Allow	Lowest	Always False
Allow	Normal	Malicious	Allow	Normal	Always False
Allow	Normal	Malicious	Allow	Higher	Always False
Allow	Normal	Malicious	Allow	Lowest	Normal
Allow	Normal	Malicious	Allow	Normal	Normal
Allow	Normal	Malicious	Allow	Higher	Normal

Table B.2 Listing of all Collaborative Groups, Part 2

Res. Avail.	Res. $	Res. Perf.	Ref. Avail.	Ref. $	Ref. Perf.
Allow	Normal	Malicious	Allow	Lowest	Always Pos.
Allow	Normal	Malicious	Allow	Normal	Always Pos.
Allow	Normal	Malicious	Allow	Higher	Always Pos.
Allow	Normal	Malicious	Allow	Lowest	Hype
Allow	Normal	Malicious	Allow	Normal	Hype
Allow	Normal	Malicious	Allow	Higher	Hype
Allow	Normal	Normal	Allow	Lowest	Always False
Allow	Normal	Normal	Allow	Normal	Always False
Allow	Normal	Normal	Allow	Higher	Always False
Allow	Normal	Normal	Allow	Lowest	Normal
Allow	Normal	Normal	Allow	Normal	Normal
Allow	Normal	Normal	Allow	Higher	Normal
Allow	Normal	Normal	Allow	Lowest	Always Pos.
Allow	Normal	Normal	Allow	Normal	Always Pos.
Allow	Normal	Normal	Allow	Higher	Always Pos.
Allow	Normal	Normal	Allow	Lowest	Hype
Allow	Normal	Normal	Allow	Normal	Hype
Allow	Normal	Normal	Allow	Higher	Hype
Allow	Higher	Malicious	Allow	Lowest	Always False
Allow	Higher	Malicious	Allow	Normal	Always False
Allow	Higher	Malicious	Allow	Higher	Always False
Allow	Higher	Malicious	Allow	Lowest	Normal
Allow	Higher	Malicious	Allow	Normal	Normal
Allow	Higher	Malicious	Allow	Higher	Normal
Allow	Higher	Malicious	Allow	Lowest	Always Pos.
Allow	Higher	Malicious	Allow	Normal	Always Pos.
Allow	Higher	Malicious	Allow	Higher	Always Pos.
Allow	Higher	Malicious	Allow	Lowest	Hype
Allow	Higher	Malicious	Allow	Normal	Hype
Allow	Higher	Malicious	Allow	Higher	Hype
Allow	Higher	Normal	Allow	Lowest	Always False
Allow	Higher	Normal	Allow	Normal	Always False
Allow	Higher	Normal	Allow	Higher	Always False
Allow	Higher	Normal	Allow	Lowest	Normal
Allow	Higher	Normal	Allow	Normal	Normal
Allow	Higher	Normal	Allow	Higher	Normal
Allow	Higher	Normal	Allow	Lowest	Always Pos.
Allow	Higher	Normal	Allow	Normal	Always Pos.
Allow	Higher	Normal	Allow	Higher	Always Pos.
Allow	Higher	Normal	Allow	Lowest	Hype

Table B.3 Listing of all Collaborative Groups, Part 3

Res. Avail.	**Res. $**	**Res. Perf.**	**Ref. Avail.**	**Ref. $**	**Ref. Perf.**
Allow	Higher	Normal	Allow	Normal	Hype
Allow	Higher	Normal	Allow	Higher	Hype
Allow	Lowest	Normal	Deny	N/A	N/A
Allow	Lowest	Malicious	Deny	N/A	N/A
Allow	Normal	Normal	Deny	N/A	N/A
Allow	Normal	Malicious	Deny	N/A	N/A
Allow	Higher	Normal	Deny	N/A	N/A
Allow	Higher	Malicious	Deny	N/A	N/A

BIBLIOGRAPHY

[1] A. Abdul-Rahman and S. Hailes, *Supporting Trust in Virtual Communities*, In 33rd Annual Hawaii International Conference on System Sciences (HICSS-33), 2000.

[2] K. Aberer, *P-grid: A Self-Organizing Access Structure for P2P Information Systems*, Cooperative Information Systems, 9th International Conference, CoopIS 2001, 2001.

[3] K. Aberer and Z. Despotovic, *Managing Trust in a Peer-To-Peer Information System*, Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management, 2001.

[4] B. Alunkal, I. Veljkovic, and G. von Laszewski, *Reputation-based Grid Resource Selection*, Argonne National Laboratory, New Orleans, Louisiana, 27 Sept. 2003. Accessed: 30 June, 2005. `http://www-unix.mcs.anl.gov/~laszewsk/papers/vonLaszewski--reputation.pdf`

[5] F. Azzedin and M. Maheswaran, *Evolving and Managing Trust in Grid Computing Systems*, IEEE Canadian Conference on Electrical & Computer Engineering (CCECE '02), May 2002.

[6] R. Butler, V. Welch, D. Engert, I. Foster, S. Tuecke, J. Volmer, and C. Kesselman, *A National-Scale Authentication Infrastructure*, IEEE Computer, 33(12):60-66, 2000.

[7] R. Buyya, D. Abramson, and J. Giddy, *Grid Resource Management, Scheduling and Computation Economy*, 2nd International Workshop on Global and Cluster Computing (WGCC 2000), Tsukuba/Tokyo, Japan, March 15 - 17, 2000.

[8] I. Clarke, O. Sandeberg, B. Wiley, and T. W. Hong, *Freenet: A Distributed Anonymous Information Storage and Retrieval System*, Workshop on Design Issues in Anonymity and Unobservability, 2000.

[9] E. Damiani, C. Vimercati, S. Paraboschi, P. Samarati, and F. Violante, *A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks*, In Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington DC, November, 2002.

[10] C. Dellarocas, *Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behavior*, Proceedings of the 2nd ACM Conference on Electronic Commerce, Oct 2000.

[11] "eBay," Web page, Accessed: 30 June, 2005. `http://www.ebay.com`.

[12] I. Foster, C. Kesselman, and S. Tuecke, *The Nexus approach to integrating multithreading and communication*, Journal of Parallel and Distributed Computing, 37:70-82, 1996.

[13] I. Foster, J. Geisler, C. Kesselman, and S. Tuecke, *Managing multiple communication methods in high-performance networked computing systems*, Journal of Parallel and Distributed Computing, 40:35-48, 1997.

[14] I. Foster, N. Karonis, C. Kesselman, and S. Tuecke, *Managing Security in High-Performance Distributed Computations*, Cluster Computing, 1(1):95-107, 1998.

[15] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, *A Security Architecture for Computational Grids*, Proc. 5th ACM Conference on Computer and Communications Security Conference, pp. 83-92, 1998.

[16] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, 1999.

[17] I. Foster, C. Kesselman, and S. Tueke, *The Anotomy of the Grid: Enabling Scalable Virtual Organizations*, International Journal of High Performance Computing Applications, 15(3): p. 200-222, 2001.

[18] D. Fudenberg and J. Tirole, *Game Theory*, ISBN 0262061414, MIT Press, 1991.

[19] "The Globus Project website," Web Page, Accessed: 30 June, 2005. `http://www.globus.org`.

[20] P. Golle, K. Leyton-Brown, and I. Mironov, *Incentives for Sharing in Peer-to-Peer Networks*, Proc. of Electronic Commerce'01, 2001.

[21] "Gnutella," Web page, Accessed: 30 June, 2005. `http://www.gnutella.com`.

[22] M. Humphrey and M. Thompson, *Security Implications of Typical Grid Computing Usage Scenarios*, In HPDC 10, August 2001.

[23] T. Johnson and P. Krishna, *Lazy Updates for Distributed Search Structure*, 1993 ACM SIGMOD International Conference on Management of Data, 1993.

[24] S. Kamvar, M. Schlosser, and H. Garcia-Molina, *The Eigentrust Algorithm for Reputation Management in P2P Networks*, In Proc. of the Twelfth International World Wide Web Conference, Budapest, Hungary, May 2003.

[25] S. Kamvar, M. Schlosser, H. Garcia-Molina, *EigenRep: Reputation Management in P2P Networks*, In Twelfth International World Wide Web Conference, 2003.

[26] "Kazaa," Web page, Accessed: 30 June, 2005. `http://www.kazaa.com`.

[27] S. Lee, R. Sherwood, and B. Bhattacharjee, *Cooperative Peer Groups in NICE*, IEEE Infocom, April 2003.

[28] S. Marsh, *Formalising Trust as a Computational Concept*, PhD Thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.

[29] D. H. McKnight and N. L. Chervany, *The Meanings of Trust*, Technical Report WP9604, University of Minnesota Management Information Systems Research Center, 1996.

[30] "Project Nice at University of Maryland," Web Page, Accessed: 30 June, 2005. `http://www.cs.umd.edu/projects/nice`.

[31] J. Novotny, S. Tuecke, and V. Welch, *An Online Credential Repository for the Grid: MyProxy*, Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.

[32] "Overstock.com Auctions," Web page, Accessed: 30 June, 2005. `http://auctions.overstock.com`.

[33] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke, *A Community Authorization Service for Group Collaboration*, Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.

[34] L. Ramaswamy, L. Liu, *Free Riding: A New Challenge to Peer-to-Peer File Sharing Systems*, 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track7, January 06 - 09, 2003 Big Island, Hawaii.

[35] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, *A Scalable Content Addressable Network*, ACM SIGCOMM, 2001

[36] A. Rowstron and P. Druschel, *Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-To-Peer Systems*, Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms, 2001.

[37] S. Saroiu, P. K. Gummadi, and S. D. Gribble, *A Measurement Study of Peer-to-Peer File Sharing Systems*, in Proceedings of Multimedia Computing and Networking (MMCN'02), 2002.

[38] W. Sears, Z. Yu, and Y. Guan, *An Adaptive Reputation-based Trust Framework for Peer-to-Peer Applications*, in Proceedings of the 4th IEEE International Symposium on Network Computing and Applications (NCA 2005), Cambridge, MA, July 27 - 29, 2005.

[39] A. Selcuk, E. Uzun, and M. Pariente, *A Reputation-Based Trust Management System for P2P Networks*, CCGRID2004: 4th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2004.

[40] S. Song and K. Hwang, *Trusted Grid Computing with Security Assurance and Resource Optimization*, in Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems (PDCS-2004), San Francisco, USA. September 15-17, 2004.

[41] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, *Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications*, ACM SIGCOMM, 2001.

[42] V. Swarup and J. T. Fabrega, *Trust: Benefits, Models, and Mechanisms*, Secure Internet Programming: Security Issues for Mobile and Distributed Objects, Lecture Notes in Computer Science, New York: Springer-Verlag, 1999.

[43] H. Yokota, Y. Kanemasa, and J. Miyazaki, *Fat-btree: An Update-Conscious Parallel Directory Structure*, 15th International Conference on Data Engineering, 1999.

[44] B. Yu and M. P. Singh, *A Social Mechanism of Reputation Management in Electronic Communities*, Cooperative Information Agents, 7th International Conference, CoopIS 2000, 2000.

[45] G. Zacharia, A. Moukas, and P. Maes, *Collaborative Reputation Mechanisms in Electronic Marketplaces*, 32nd Annual Hawaii International Conference on System Sciences (HICSS-32), 1999.

ACKNOWLEDGEMENTS

I would like to say thank you to everyone who believed in me. Fall of 2002 began my adventure as a graduate student here at Iowa State University after the company I was working for decided to leave town. I must say I am fortunate to have met what it would seem all of the right people at the right time.

First of all I would like to thank Dr. Yong Guan for taking me in under his wing and providing me the support I needed to get accepted into the graduate college. One of my first classes was CPRE 592, which coincidentally was also Dr. Guan's first class as a professor. It was during that first semester that I approached Dr. Guan and asked if he could be my advisor, which of course he said yes!

I would also like to thank Dr. Thomas Daniels for being another one of my major professers. Dr. Daniels taught my CPRE 531 and CPRE 534 classes and has been an excellent person to talk to at any time. Interesting fact: Dr. Daniels can brew his own beer and apparently it is very good!

Thirdly I would like to thank Dr. Clifford Bergman for electing to be on my committee. Dr. Dergman taught CPRE 533 which is about cryptography and one of the more interesting and challenging subjects I have ever studied. It was very hard not to stay attentive in his classes as they were always interesting.

I would also like to thank my wife Leona, who has been so very patient with me these last three years. Originally we were to be married when she graduated school, yet circumstances apparated that allowed me to go back to school and pursue a Master's Degree. Hence, I was the one in school when we were married. I bet she will be glad to

have me back working full time on housework!

Much gratitude is made to John, Margo, and Lisa, my parents and sister, for all of the support and encouragement they have given me through my term here at Iowa State University. If it were not for all of my family, I do not know how I would have made it.

Thank you to Zhen Yu for helping me with several aspects of this thesis. Our discussions were very enlightening, and I am glad you were there to help me when I needed it.

A show of appreciation to my coworkers at Ames Laboratory. The knowledge I have received from my time as a student employee and fellow staff has just been incredible!

A final thank you to Pam Meyers and staff in the Department of Electrical and Computer Engineering Student Services for providing me guidance with the ton of paperwork that needed to be filled out.